1.The sieve of Eratosthenes Algorithm

Problem statement: The sieve of Eratosthenes algorithm is an ancient algorithm that is used to find all the prime numbers less than given number N. It can be done using O(n*log(log(n))) operations.

Example:

1. Print all prime numbers less than 15.
2. Create list = 2,3,4,5,6,7,8,9,10,11,12,13,14,15
3. num=2.
4. traverse from 2 to √15.
5. num =2. Remove all multiples of 2 then list= [2,3,0,5,0,7,0,9,0,11,0,13,0,15]
6. num=3, immediate non zero number.
7. Mark all multiples of 3 then list= [2,3,0,5,0,7,0,0,0,11,0,13,0,0]
8. num=5, which is the next prime, but num is not <=√15.
9. Now we have generated all the prime numbers less than 15. Prime numbers less than 15 are 2, 3, 5, 7, 11, 13.

Explanation:

We create a list of all numbers from 2 to 15.

2	3	4	5	6	7	8	9	10	11	12	13	14	15

According to the algorithm we will mark all the numbers which are divisible by 2 and are greater than or equal to the square of it.

2	3	4	5	6	7	8	9	10	11	12	13	14	15

Now we move to our next unmarked number 3 and mark all the numbers which are multiples of 3 and are greater than or equal to the square of it.

2	3	4	5	6	7	8	9	10	11	12	13	14	15

The final list is:

2	3	4	5	6	7	8	9	10	11	12	13	14	15

So the prime numbers are the unmarked ones: 2, 3, 5, 7, 11, 13,

Algorithm:

1. Generate numbers from 2 to N (2 is the smallest prime).

2. Traverse from smallest prime number which is num = 2.

3. Eliminate or mark all the multiples of num (as 0 or -1) which are lesser than or equal to T. It will help remove non-prime numbers and will help to reduce our number of comparisons to check for each number.

4. Update the value of num to the immediate next prime number. The next prime is the next (non 0 or -1) number in the list which is greater than the current number (num).

5. Repeat step three until num<=√N.
6. Traverse the whole list and number print. All the numbers (>=0) will be our required prime numbers lesser than N (given number).

Reason for traversing until (√N): here will not exist any factor of N greater than (√N). Suppose x and y are the factors of N. In that case, x*y = N . Hence, at least one or both should be <=√ N, so we need to traverse until (<=√ N) only.

Psudo Code:

```
  boolean list = {
         2 to N
  }
 for i = 2 to <= √N
        if (list[i])
             then {
        for j = i to j * i < N
        list[j * i] = false // Eliminate the multiple of i
 }
End
//To print Primes
  For i = 2 to N
   if (list[i])
         print(i + 1)
End
```

Source Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

void SieveOfEratosthenes(int N)
{


    bool prime[N + 1];
    memset(prime, true, sizeof(prime));

    for (int p = 2; p * p <= N; p++) {

        if (prime[p] == true) {

            for (int i = p * p; i <= N; i += p)
                prime[i] = false;
        }
    }


    for (int p = 2; p <= N; p++)
        if (prime[p])
            cout << p << " ";
}
```

```
int main()
{
    int N = 15;
    cout << "Following are the prime numbers smaller "
         << " than or equal to " << N << endl;
    SieveOfEratosthenes(N);
    return 0;
}
```

Output:

```
Following are the prime numbers smaller  than or equal to 15
2 3 5 7 11 13
Process returned 0  (0x0)   execution time : 0.907 s
Press any key to continue.
```

Time Complexity:

Time Complexity: O(n*log(log(n)))

1. **Time required to eliminate numbers is constant which is (n/2 + n/3 + n/4 +n/5.............N)**

2. **Take** N **common from above equation will be n*(1/2 + 1/3 + 1/4 + 1/5.............N)**

3. **Taking the harmonic progression of prime numbers will be (1/2 + 1/3 + 1/4 + 1/5.............=log(log(n))**

4. **Hence the complexity is O(n*log(log(n)))**